Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Ernest Ivnik

Indeksiranje in iskanje po simboličnih glasbenih zbirkah

diplomsko delo

univerzitetni študijski program računalništva in informatika

Mentor: doc. dr. Matija Marolt

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika dela:

V diplomskem delu preučite sisteme za iskanje po simboličnih glasbenih zbirkah. Analizirajte algoritme za iskanje kot tudi uporabniške vmesnike, ki se največkrat uporabljajo za podajanje povpraševanj. Implementirajte lastni sistem za indeksiranje in iskanje na podlagi Galoisovih polj in ga ovrednotite.

Mentor: doc. dr. Matija Marolt

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Ernest Ivnik, z vpisno številko **63070021**, sem avtor diplomskega dela z naslovom:

Indeksiranje in iskanje po simboličnih glasbenih zbirkah

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matije Marolta,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 15. aprila 2014 Podpis avtorja:

Zahvaljujem se doc. dr. Matiji Maroltu za usmeritev in pomoč pri izdelavi diplomske naloge. Zahvaljujem se tudi družini za podporo v času študija.

Kazalo

Povzetek

V diplomski nalogi je realiziran eden od pristopov indeksiranja in iskanja po simboličnih glasbenih zbirkah. Na podlagi obstoječih spletnih aplikacij je bil sestavljen uporabniški vmesnik, ki podpira tekstovni vnos in vnos s klaviaturo ter iskanje po melodiji in ritmu. Indeksiranje in iskanje temeljita na indeksu melodičnih podpisov ter n-gramih. Sestavljeni sta bili spletni strani za iskanje, prikaz rezultatov in stran za posodobitev zbirke. Za realizacijo je bil izbran jezik Java, spletni strežnik Tomcat, spletne strani pa so vsebovale HTML in JavaScript. Za zvočno in grafično predstavitev not so bile uporabljene obstoječe knjižnice. Realizirane so bile metode za zajem podatkov, indeksiranje, točno in približno iskanje, vnos iskanja in prikaz rezultatov. Realizacija je bila testirana na dejanski zbirki. Točno iskanje je bilo hitrejše od približnega pri daljših iskalnih nizih. Ostali vhodni parametri, razen iskanje samo začetka zapisov, pa niso poglavitno vplivali na čas izvajanja.

Ključne besede: simbolični glasbeni zapisi, indeksiranje, iskanje, n-grami, melodični podpisi

Abstract

This thesis contains realization of a single approach to indexing and searching in symbolic music collections. On the basis of existing web applications the user interface was built that supports text and keyboard input with search by melody and rhythm. Indexing and searching are based on a melodic signature index and n-grams. Web pages were created for search input, presentation of results and a page that provides collection update. Java and Tomcat web server were chosen for the realization. Web pages contain HTML and JavaScript. Existing libraries were used for sound and graphic presentation of notes. The methods realized were for data capture, indexing, exact and approximate search, search input and display of results. The realization has been tested on an actual collection. Exact search was faster than approximate search for long search strings. Other input parameters, except for the one limiting search to the beginning of records didn't have major influence on the time of search.

Keywords: symbolic music notation, indexing, search, n-grams, melodic signatures

Poglavje 1

Uvod

V današnjih časih uporaba računalnikov in interneta vse bolj narašča na vseh področjih delovanja. Eno od teh področij je tudi glasba.

Vsak se je verjetno že srečal s popularnimi zvočnimi zapisi, kot so na primer mp3 in wav. Manj znano pa je področje simboličnih glasbenih zapisov pri katerih je najbolj znan zapis MusicXML. Ti v večini vsebujejo notne zapise v takšnem ali drugačnem formatu. Tudi število takih zapisov se je v zadnjem času močno povečalo. K temu so pripomogli različni projekti pretvarjanja starih papirnatih notnih zapisov v elektronsko obliko. Prav tako se večina glasbenikov oziroma skladateljev danes poslužuje raznih računalniških programov za skladanje in pisanje glasbe, ki v prvi vrsti podpirajo tudi simbolično zapisovanje glasbe.

Osnovno iskanje po imenu avtorja ali naslovu skladbe včasih ne zadostuje, ker se lahko zgodi, da preprosto poznamo samo del melodije ali ritma in bi želeli najti zapis te skladbe. Zato se je za iskanje simboličnh glasbenih zapisov dobro obrniti tudi na njihove vsebinske lastnosti. Nekatere glavne lastnosti so višina tonov (melodija) in trajanje tonov (ritem).

Tako iskanje pa sproži vrsto drugih problematik. Glavna od teh je, kako najučinkoviteje razvrstiti uporabljene lastnosti, da so čim hitreje dostopne. Pred tem je dobro razmisliti, katere je smiselno uporabiti in kako podrobne naj bodo, da bo samo iskanje po njih čim hitrejše in učinkovitejše. Zaradi

naraščajoče količine simboličnih glasbenih zapisov je zato vse večja potreba po učinkovitih metodah indeksiranja in iskanja teh zapisov po njihovih vsebinskih lastnostih.

Na svetovnem spletu že obstajajo nekatere spletne aplikacije in programčki, ki na podlagi lastnosti simboličnega glasbenega zapisa poiščejo ustrezne zapise. Žal je večina teh, kot bomo videli v nadaljevanju v pomanjkljivem, poskusnem stanju, nadaljnji razvoj pa je zelo počasen ali v mirovanju.

V upanju, da bodo posamezne tehnike iskanja in indeksiranja postale malo bolj znane, smo v tej nalogi sami poskusili realizirati enega od trenutno dokumentiranih pristopov. Za realizacijo indeksiranja in iskanja smo izbrali indeks melodičnih podpisov (angl. Melodic Signature Index - MSI). Zanj smo se po premisleku odločili zato ker nudi:

1. hitro točno iskanje; poseben pristop, pri katerem dolžina iskalnega niza ne vpliva na samo hitrost in zahtevnost iskanja,

2. vrednosti indeksa se računajo iz razlike med višinami sosednih tonov v n-gramu. Glavna razpoznavna lastnost je melodija,

3. nudi dokaj enostavno možnost nadgradnje osnovnega iskanja po melodiji z bolj podrobnimi lastnostmi in parametri,

4. preprosto teoretično in matematično ozadje, ki temelji na algebraičnih podpisih in Galoisovem polju, pri katerem ni potrebno poglobljeno poznavanje glasbene teorije.

Pred začetkom naše realizacije smo si pogledali najbolj pogoste vrste iskanja in načine vnosa poizvedb za iskanje pri trenutno obstoječih spletnih aplikacijah in programčkih. Kot bomo videli v nadaljevanju, so vrste iskanja precej enovite. Nekateri načini vnosa poizvedbe so tudi precej nenatančni in za iskanje praktično neuporabni, saj je vnos preveč odvisen od naključja. Naš uporabniški vmesnik smo glede na videno in po naši presoji in zmožnosti poskušali narediti čim bolj prijazen uporabniku, tako da bo nudil zadosten

nivo nadzora nad vnosom željene poizvedbe za iskanje, a hkrati ne bo preveč zapleten za preprostega uporabnika.

V nadaljevanju naloge si bomo najprej podrobneje ogledali naš realiziran primer z vidika, kot ga doživlja uporabnik, in še s tehničnega vidika, kjer se bomo malo bolj poglobili v matematične osnove, na katerih temelji, in samo delovanje naše rešitve.

Na koncu nas bo zanimalo še, kako se naša realizacija obnese na dejanskih podatkih. Pri tem nas bo posebej zanimal čas, ki je potreben za določeno iskanje, in dejavniki, ki vplivajo nanj.

Poglavje 2

Obstoječe spletne aplikacije

Na svetovnem spletu obstaja kar nekaj spletnih aplikacij z namenom iskanja po simbolnih glasbenih zapisih. Izmed teh smo odkrili in si pogledali naslednje:

1. Neuma [1]
2. Peachnote [2]
3. Musipedia [3]
4. Melodycatcher [4]
5. Hymnary [5]
6. Folktunefinder [6]
7. Nzdl [7]
8. DoDoSoSo [8]
9. Tunefind [9]
10. Themefinder [10]

Ogledali si bomo, katere vrste iskanj so podprte s strani katere izmed naštetih spletnih aplikacij, in omenili nekaj dobrih in slabših primerov načina vnosa

poizvedbe iskanja pri določeni vrsti iskanja. Zanimal nas bo tudi način prikaza in vsebina rezultatov.

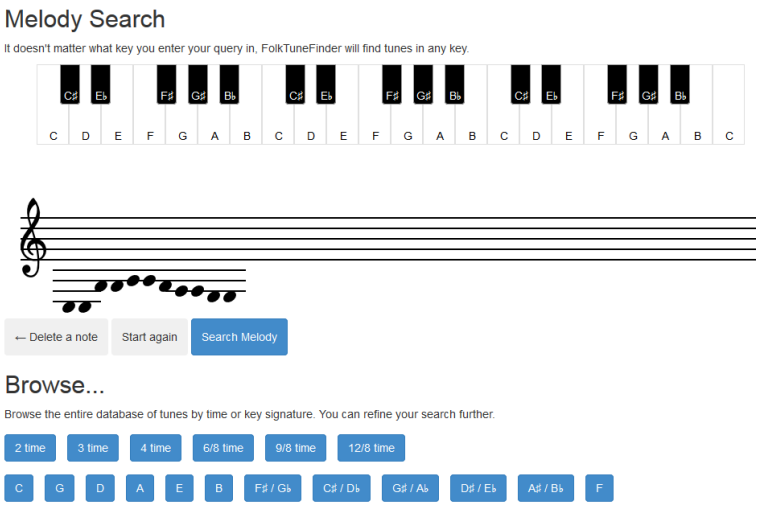
2.1 Vrste iskanja

Vse zgornje spletne aplikacije omogočajo iskanje po melodiji. Velikokrat se ta vrsta iskanja uporablja skupaj z ritmom ali pa posamezne spletne aplikacije dodatno ponujajo še iskanje s tehniko melodičnih obrisov.

2.1.1 Iskanje po melodiji

Med ogledanimi spletnimi aplikacijami, ki podpirajo samo iskanje glede na melodijo, sta na primer Melodycatcher in DoDoSoSo, ki sta sestavljeni na zelo preprost način. Pri prvi uporabnik lahko vnaša iskalno poizvedbo s klaviaturo ali tekstovno, preprosto tako, da vpiše ime note, druga pa podpira le vnos zlogov, ki predstavljajo določene višine.

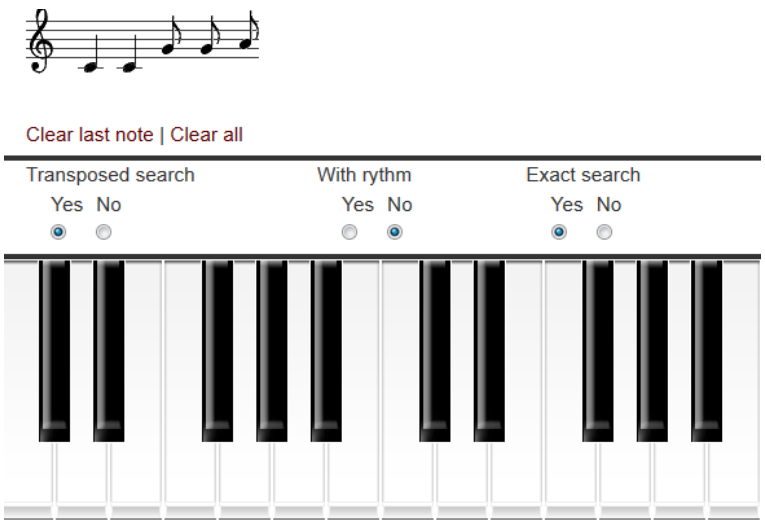
Malo bolj prefinjen primer predstavlja Folktunefinder, ki sicer enako omogoča le iskanje po melodiji, vendar ima to posebnost, da uporabnik iskanje lahko omeji na manjše dele celotne zbirke zapisov, ki ustrezajo določenim taktovskim načinom, tonalitetam ali inštrumentom. Možno je tudi samo brskanje s temi parametri samimi brez vnosa melodije. Uporabniški vmesnik za iskanje je prikazan na Sliki 2.1.



Slika 2.1: Uporabniški vmesnik za iskanje spletne aplikacije Folktunefinder.

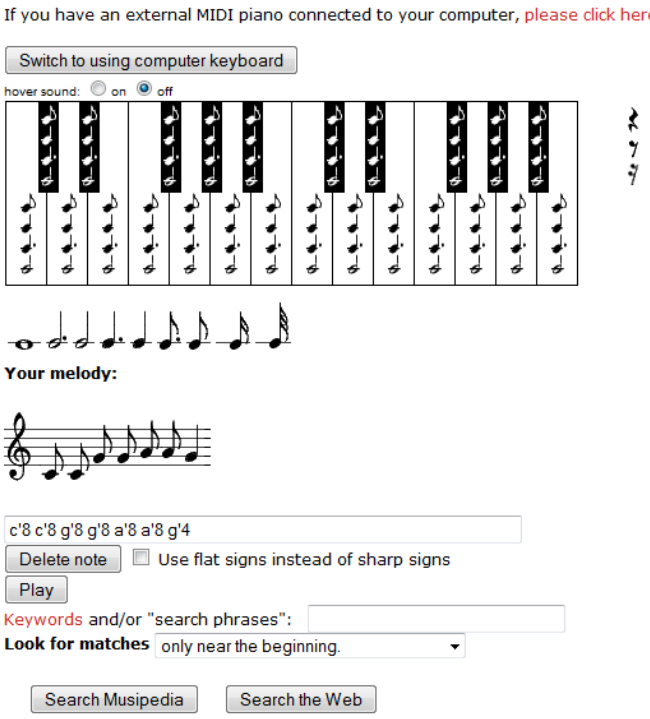
2.1.2 Iskanje po melodiji in ritmu

Velika večina ogledanih spletnih aplikacij vsebuje tako iskanje po melodiji kot po ritmu, pri čemer večji poudarek daje na melodijo. Dober predstavnik teh je Neuma. Tu je melodija glavna lastnost, po kateri se išče, ritem pa zgolj sekundarna lastnost, ki jo primerja šele, ko je melodija že preverjena. Poleg same poizvedbe ima uporabnik možnost določiti še, ali naj se v iskanje vključi tudi zapise, ki se ujemajo, če melodijo transponiramo ali primerjamo podobnost z ritmom. Izbira lahko med točnim in približnim iskanjem. Kljub dobri razdelanosti parametrov iskanja pa ji manjka dober uporabniški vmesnik. Kot vidimo na Sliki 2.2, nam Neuma nudi samo en način vnosa poizvedbe za ritem in melodijo, in sicer z virutalno klaviatuo. Trenutne vrednosti poizvedbe so prikazane na notnem črtovju, kar pa ni povsem dovolj za dobro predstavo o vnešeni melodiji ali ritmu. Dobrodošla bi bila še tekstovna, zvočna predstavitev in več interaktivnosti. Ravno tako je vnos ritma omejen zgolj na nekaj osnovnih trajanj, ki jih je težko vnesti le s pomočjo klaviature.



Slika 2.2: Uporabniški vmesnik spletne aplikacije Neuma.

Še boljši primer strani, ki v celoti podpira iskanja po melodiji in ritmu, je Musipedia, kjer lahko posebej iščemo tudi samo po ritmu. Nudi še vmesno nastavitev občutljivosti za melodijo in ritem, kjer lahko nastavimo večjo ali manjšo pomembnost enega ali drugega. Prav tako Musipedia nudi najboljše in najbolj raznolike načine vnosa melodije in ritma. Najprej je tu vnos s pomočjo virtualne klaviature. Na vsaki tipki je možno izbrati tudi dolžino note, kot je prikazano na Sliki 2.3. Zaigrane note lahko nato še tekstovno urejamo. Tekstovni zapis je precej preprost in je sestavljen iz imen not in trajanj, ki so predstavljena s številko. Podpira tudi priključitev na klaviaturo midi in vnos poizvedbe prek nje. Ritem se lahko tapka na tipkovnico. Možno je tudi iskanje s petjem v mikrofonom. Čeprav se zadnje sliši kot zelo prikladen način vnosa poizvedbe za iskanje, je praktično neuporaben zaradi zelo velike nenatančnosti in naključnosti.



Slika 2.3: Uporabniški vmesnik za enega od načinov vnosa pri spletni aplikaciji Musipedia.

Nazadnje velja posebej omeniti tudi Peachnote, ki za iskanje lahko uporablja kombinacijo točnih višin tonov, melodije, ritma ali celo akordov in temelji na n-gramih.

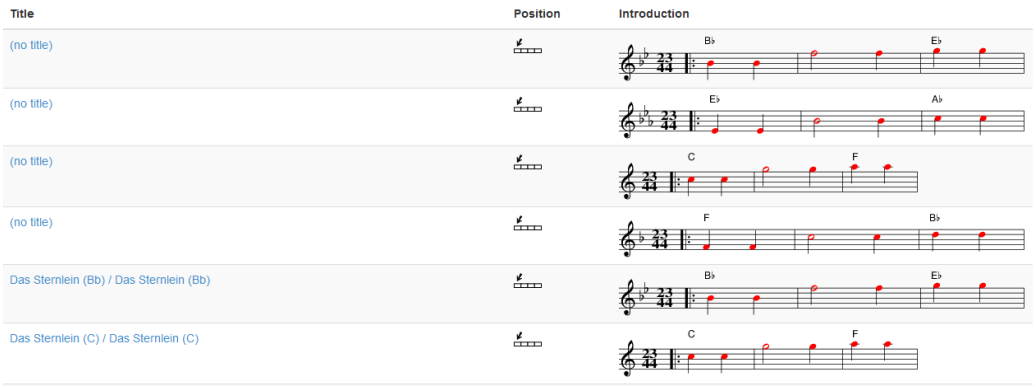
Ostale, ki podpirajo to vrsto iskanja, so še Nzdl, Hymnary in Themefinder, vendar pa se posebno ne odlikujejo.

2.1.3 Iskanje po melodičnih obrisih

Iskanje je možno tudi s tehniko melodičnih obrisov (sprememba višine tona skozi čas) [11]. Zanjo se uporablja posebna vrsta zapisa, ki se ji reče Parsonova koda (angl. Parsons code) [12]. Tak način iskanja je med drugim možen pri spletnih aplikacijah Musipedia, Tunefind in Themefinder.

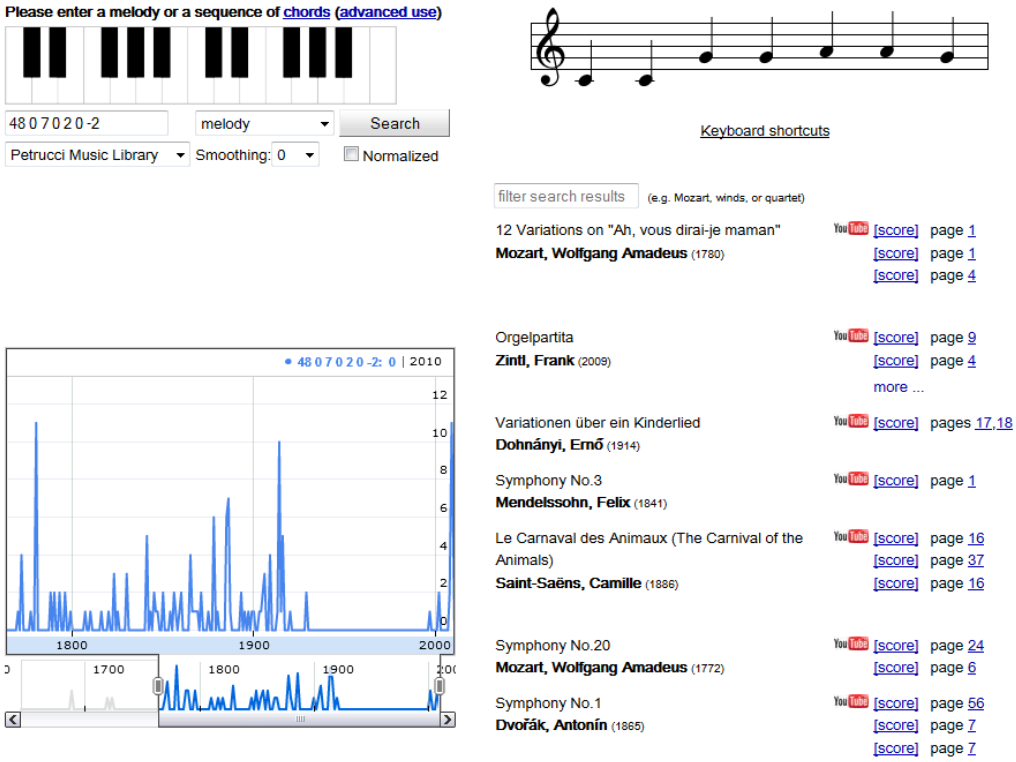
2.2 Predstavitev rezultatov

Večina ogledanih spletnih aplikacij nudi prikaz osnovnih podatkov o naslovu in avtorjih posameznega rezultata. Možen je tudi prenos tako datoteke midi kakor tudi simboličnega glasbenega zapisa. Nekatere, kot je naprimer Musipedia, nudijo tudi relevantne povezave na zunanje spletne strani, kot sta Youtube in Amazon. Posebno velja omeniti spletno aplikacijo Folktunefinder, kjer je vsak zadetek zelo nazorno prikazan z izsekom notnega zapisa. Dele, ki se ujemajo z našo poizvedbo, posebej obarva, kot lahko vidimo na Sliki 2.4.



Slika 2.4: Prikaz rezultatov, ki ga nudi Folktunefinder.

Dobro delitev rezultatov iskanja ima tudi Peachnote. Rezultati iskanja so tam prikazani na grafu po letih. Ta graf lahko uporabnik normalizira ali zgladi. Interaktivno se lahko izbere določen časovni interval in notne zapise, vsebovane v njem, kar si lahko ogledamo na Sliki 2.5.



Slika 2.5: Uporabniški vmesnik za iskanje skupaj s prikazom rezultatov pri spletni aplikaciji Peachnote.

2.3 Lastna implementacija

Pri zasnovi lastnega uporabniškega vmesnika smo najprej preučili zbrane podatke o obstoječih spletnih aplikacijah, ki smo jih omenili. Ob opazovanju in preizkušanju omenjenih spletnih aplikacij smo, kot je to prikazano v Tabeli 2.1, opazili, da je bil najbolj pogost način vnosa poizvedbe za iskanje tekstovni način. Nato ji je sledil vnos s pomočjo klaviature.

Spletne aplikacija/način vnosa	Tekstovno	Klaviatura - melodija	Klaviatura - ritem	Tipkovnica - tapkanje ritma	Mikrofon	Notno črtovje
NEUMA		x	x			
PEACHNOTE	x	x				
MUSIPEDIA	x	x	x	x	x	
MELODYCATCHER	x	x				
HYMNARY		x				
FOLKTUNEFINDER	x	x				
NZDL		x				
DODOSOSO	x					x
TUNEFIND	x					
THEMEFINDER	x					

Tabela 2.1: Načini vnosa, ki ga uporabljajo izbrane spletne aplikacije.

Velikokrat sta bila ta dva vnosa uporabljena tudi tako, da sta se med seboj dopolnjevala. Za našo realizacijo smo izbrali oboje, saj klaviatura poskrbi za hiter in enostaven vnos melodije, s tekstovnim vnosom pa lahko dodatno popravimo in izoblikujemo našo poizvedbo, da ustreza našim željam. Za preprostega uporabnika tako niti ni nujno poglabljanje v tekstovne zapise melodije, bolj napreden uporabnik pa ima dovolj možnosti, da podrobno definira svojo iskalno poizvedbo.

Glavna lastnost po kateri je bilo opravljeno iskanje pri vseh ogledanih aplikacijah, je bila melodija. Nekatere so imele poleg tega še opcijo iskanja po ritmu, kar prikazuje Tabela 2.2. Ostalih vrst iskanja je bilo zanemarljivo malo. Pri naši realizaciji smo zato uporabili melodijo kot glavno lastnost iskanja. Ritem kot manj pomemben pa smo postavili kot možno dodatno lastnost.

Spletna stran/Lastnost iskanja	Melodija	Ritem	Melodični obrisi
NEUMA	x	x	
PEACHNOTE	x	x	
MUSIPEDIA	x	x	x
MELODYCATCHER	x		
HYMNARY	x	x	
FOLKTUNEFINDER	x		
NZDL	x	x	
DODOSOSO	x		
TUNEFIND	x		x
THEMEFINDER	x		x

Tabela 2.2: Lastnost, po kateri je narejeno iskanje zapisov pri določeni spletni aplikaciji.

Da si lahko uporabnik lažje predstavlja trenutno vnešene note oziroma tone je seveda nujno, da posamezne tipke klaviature vsebujejo ustrezne zvočne tone. Poleg tega predstavo olajša tudi možnost predvajanja celotne melodije vnešene poizvedbe. Oboje smo zato vključili v našo implementacijo. Dodali smo ji še prikaz vnešenih tonov na notnem črtovju.

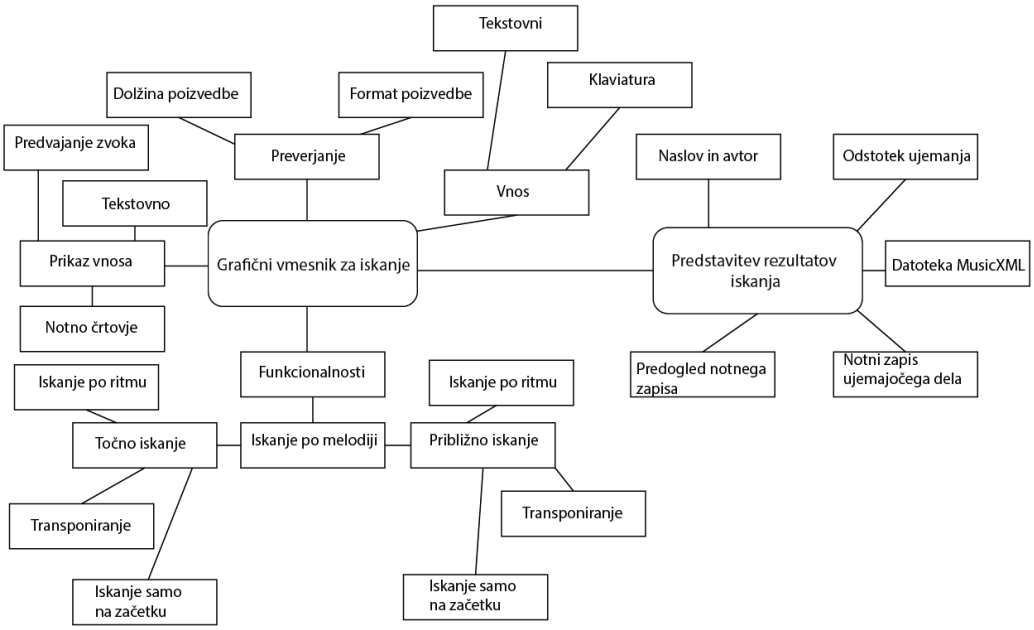
Za prikaz rezultatov v našem primeru smo želeli, da uporabnik izve čim več informacij iz posameznega rezultata, tako da smo vklučili prikaz izseka notnega zapisa, kjer se rezultat ujema z iskalno poizvedbo, in možnost predogleda celotnega notnega zapisa. Hkrati smo uporabniku omogočili tudi takojšni prenos rezultata v izvornem formatu, ki je bil v našem primeru MusicXML.

Poglavje 3

Uporabniški vidik

Naša realizacija je namenjena uporabnikom, ki vsaj bežno poznajo glasbeno teorijo, saj gre tu za iskanje notnih zapisov. Navadnemu uporabniku ti ponavadi ne pomenijo veliko. Ciljni uporabniki so torej glasbeniki ali amaterji, ki so že igrali na kakšen inštrument ali poznajo osnove glasbene teorije. V okviru uporabniškega vidika si bomo najprej pogledali funkcionalnosti, ki nam jih nudi uporabniški vmesnik za iskanje, nato pa še za prikaz rezultatov. V nadaljevanju si bomo ogledali še stran za vzdrževanje podatkov.

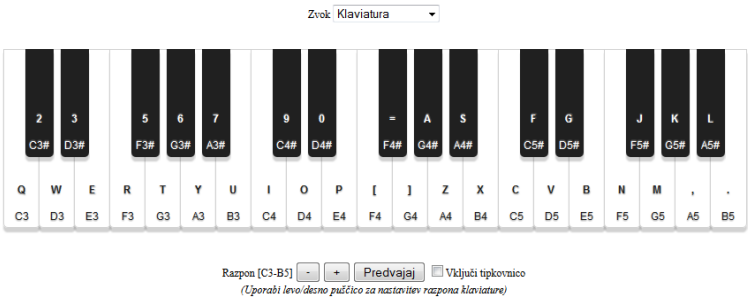
Za lažjo predstavo si na Sliki 3.1 najprej oglejmo kratko predstavitev, kaj lahko uporabnik od naše realizacije pričakuje.



Slika 3.1: Celovit pregled nad uporabniškim vmesnikom in funkcionalnostmi, ki jih nudi uporabniku.

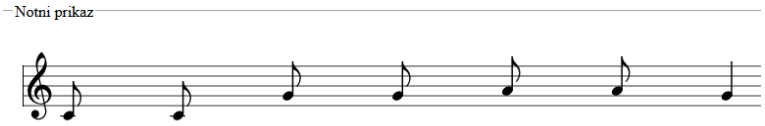
3.1 Stran za iskanje

Stran za iskanje je oblikovana tako, da je čim bolj preprosta in pregledna za uporabnika. Za začetek iskanja lahko uporabnik klika na virtualno klaviaturo, ki je prikazana na Sliki 3.2. S tem vnaša posamezne višine tonov, ki se bodo uporabile pri iskanju. Pritisk na vsak ton povzroči, da izbrani ton zazveni. Uporabnik ima nad klaviaturo v padajočem meniju na izbiro nekaj različnih glasbil, ki jih lahko uporabi za zvok tipk. Poleg tega lahko spremeni razpon oktav na klaviaturi, glede na to, katere tone želi vnesti s pritiskom na gumba "–"in "+". Uporabnik si lahko vneseno melodijo tudi zaigra s pritiskom gumba za predvajanje.



Slika 3.2: Klaviatura za vnos poizvedbe in gumbi za nastavitve razpona oktav, vključitev tipk tipkovnice in izbira zvoka tipk.

Pri vnašanju si lahko uporabnik ogleduje vnesene tone na notnem črtovju, kot prikazuje Slika 3.3.

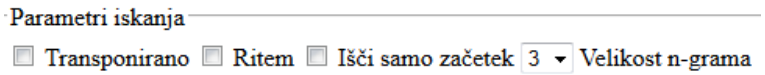


Slika 3.3: Prikaz vnesenih tonov na notnem črtovju.

V nadaljevanju na Sliki 3.4 vidimo, da stran uporabniku nudi tudi podrobnejši tekstovni vnos melodije in ritma. V tekstovno polje lahko uporabnik vnese ali iz njega izbriše posamezne tone in trajanja ali popravlja že prej vnesene tone s klaviaturo. Stran uporabniku nudi tudi pomoč pri seznanitvi s pravilnim formatom tekstovnega vnosa. Vnos se sproti preverja in vsak nepravilen niz se izpusti, upošteva pa se le pravilne. Ti so tudi prikazani na notnem črtovju, tako da uporabniku ni potrebno skrbeti za morebitne napake.

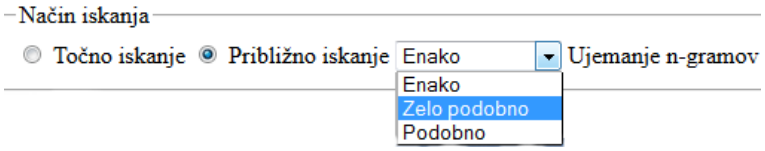
Slika 3.4: Primer tekstovnega vnosa poizvedbe.

Ko uporabnik zaključi z vnosom melodije in ritma, lahko nastavi še dodatne parametre iskanja, kot vidimo na Sliki 3.5. Izbere lahko iskanje z ritmom ali brez, transponiranje not in iskanje le po začetkih zapisov. Izbira lahko tudi med velikostjo n-gramov, ki se uporabljajo pri iskanju in indeksiranju. Privzeta velikost je tri.



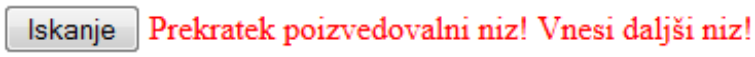
Slika 3.5: Parametri iskanja.

Uporabnik lahko odloči, kateri način iskanja bo izbral. Privzeti način je točno iskanje. Na Sliki 3.6 vidimo, da kadar uporabnik izbere približni način iskanja, mu stran ponudi še možnost izbire med stopnjami natančnosti pri iskanju.



Slika 3.6: Načini iskanja.

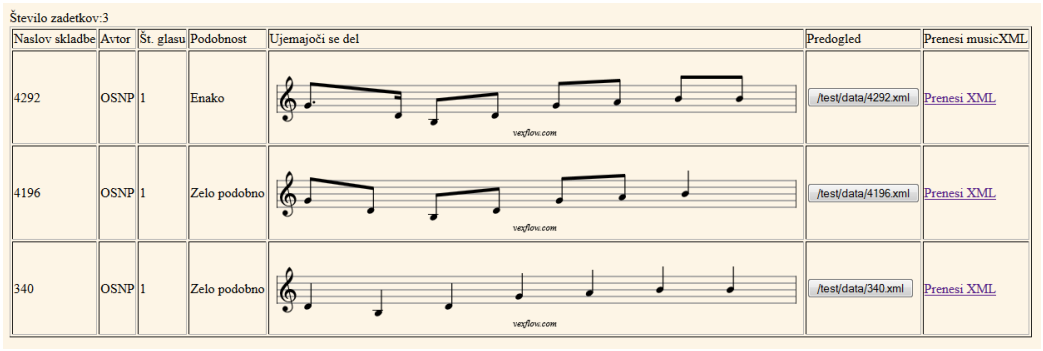
Na koncu, ko uporabnik pritisne na gumb za iskanje, ga stran obvesti o morebitni prekratki iskalni poizvedbi na način, kot je prikazan na Sliki 3.7. Drugače se poizvedba skupaj z vsemi parametri uspešno posreduje naprej in začne se iskanje.



Slika 3.7: Obvestilo v primeru napake.

3.2 Stran za prikaz rezultatov

Ta stran uporabniku nudi poleg samega pregleda rezultatov iskanja tudi nekaj drugih bolj podrobnih funkcionalnosti. Osnovni podatki, ki so uporabniku vidni že na začetku, so naslovi del, imena avtorjev in številke glasu, kjer se je zgodilo ujemanje. Temu sledi še informacija o stopnji ujemanja z našim iskalnim nizom. Poleg tega vsebuje vsak rezultat tudi prikaz približnega notnega zapisa kraja ujemanja. Izgled strani z rezultati prikazuje Slika 3.8.



Slika 3.8: Prikaz rezultatov iskanja.

V nadaljevanju lahko uporabnik odpre novo okno s predogledom celotnega notnega zapisa. Možen je tudi ogled in prenos datoteke MusicXML.

3.3 Stran za vzdrževanje zbirke

Stran s preprostim pritiskom na gumb nudi posodobitve indeksnih in ostalih datotek, da odražajo trenutno stanje zbirke. Potrebujemo jo v primeru spreminjanja, izbrisa ali dodajanja novih zapisov v zbirko. V prvi vrsti je namenjena skrbniku, vendar trenutno lahko do nje dostopa vsak uporabnik.

Poglavje 4

Temeljna teoretična podlaga lastne realizacije

Metode indeksiranja in iskanja glasbenih zapisov se ločijo predvsem po glavni lastnosti za določevanje podobnosti. Prav tako ločimo metode, ki obravnavajo zgolj značilnosti enega glasu (enoglasne), in metode, ki obravnavajo več glasov ali tonov skupaj. Slednje so uporabne, če imamo veliko akordov ali glasovi niso posebej ločeni. Med izbiranjem osnovne teoretične podlage za našo realizacijo smo si pogledali nekaj pristopov.

Srečali smo se s pristopom, ki je primeren za glasbene zapise s pogostim večglasjem ter akordi in se opira na lastnosti harmonije. Podobnost se meri v razdalji tonske višine koraka (angl. Tonal pitch step distance - TPS). TPS pa je razdalja med dvema akordoma. Manjša, kot je ta razdalja, bolj sta si dva akorda harmonično podobna [29]. Prebrali smo si tudi članek, kjer je iskanje temeljilo na euklidski ravnini in višini tonov. Ta pristop uporablja geometrično predstavitev celotnega glasbenega zapisa, kjer pari višin tonov in časa pojavitve predstavljajo točke v euklidski ravnini. Ker nas na tej predstavitvi zanimajo zgolj skupne točke z iskalno poizvedbo ali njihove translacije, nas ostale točke ne motijo. Zato je predvsem primeren za zapise z močno prisotnostjo večglasja. S to tehniko je možno iskati tudi delna ujemanja in ujemajoče podnize [31].

V nadaljevanju si bomo pogledali teorijo našega izbranega pristopa. Glavna lastnost, po kateri merimo podobnost, je melodija. Primerna je predvsem za enoglasne zapise ali zapise, ki imajo posamezne glasove ločene. Indeks, po katerem poteka iskanje, je zgrajen iz algebraičnih podpisov posameznih n-gramov. Ti podpisi združujejo vrednosti razdalj med tonskimi višinami v posameznem n-gramu v eni sami vrednosti. Takemu indeksu pravimo indeks melodičnih podpisov (angl. Melodic Signature Index – MSI). Iskanje podobnosti poteka s primerjanjem n-gramov dveh nizov. Več, kot je različnih n-gramov v enem ali drugem nizu, večja je razdalja in manj sta si dva niza podobna [30]. Preden se lotimo opisa delovanja naše realizacije, si torej najprej poglejmo nekaj teoretičnih matematičnih pojmov, na katerih ta temelji.

4.1 Galoisovo polje

Galoisovo polje (angl. Galois field – GF) je polje, ki vsebuje končno število elementov. Od tu tudi ime končno polje. To je vedno oblike p^n, kjer je p naravno število in mu rečemo karakteristika polja, n pa pozitivno celo število. Za vsak p in n obstaja torej končno polje z p^n elementi. Aritmetika v Galoisovem polju je drugačna od navadne celoštevilske, saj imamo omejeno število elementov polja. Rezultat kakršnih koli operacij je vedno element, ki je v tem polju. Za primere računanja svetujemo dodatno branje [13].

4.2 Algebraični podpisi

Za izračun algebraičnih podpisov (angl. Algebraic Signatures – AS) smo uporabljali Galoisovo polje velikosti 2^8, kar pomeni, da so elementi v GF 8-bitni. V našem primeru smo za elemente GF vzeli zaporedje dogodkov, ki jih opisuje deskriptor $D = e_0, e_1, \ldots, e_{M-1}$. M je število vseh dogodkov v zaporedju, e pa posamezen dogodek, ki predstavlja bodisi višino, trajanje tona ali razdalji med višinami dveh sosednih tonov. Definicija 4.1 predstavlja osnovno formulo algebraičnega podpisa za celoten deskriptor D. Potrebovali

smo še dodatne izpeljave tega osnovnega podpisa, ker nas ni zanimala vrednost celotnega deskriptorja, ampak samo deli. Potrebovali smo kumulativni algebraični podpis (angl. Cumulative Algebraic Signature – CAS), prikazan v Definiciji 4.2, delni algebraični podpis (angl. Parital Algebraic Signature – PAS) iz Definicije 4.3 in algebraični podpis n-grama (angl. The N-gram Algebraic Signature – NAS), ki smo ga izračnali po Definiciji 4.4.

Definicija 4.1 *Naj bo α primitivni element GF. Potem velja, da izračunamo algebraični podpis deskriptorja D po enačbi*

$$AS_\alpha(D) = e_0 + e_1 \cdot \alpha + e_2 \cdot \alpha^2 + ... + e_{M-1} \cdot \alpha^{M-1}. \tag{4.1}$$

Definicija 4.2 *Naj bo $l \in [0, M-1]$ poljuben odmik v D. Kumulativni algebraični podpis (CAS) je algebraični podpis predpone deskriptorja D, ki se konča pri elementu e_l in velja, da je*

$$CAS(D, l) = AS(e_0, \ldots, e_l). \tag{4.2}$$

Definicija 4.3 *Delni algebraični podpis (PAS) je algebraični podpis dela deskriptorja D od elementa $e_{l'}$ do elementa e_l, za katerega velja, da je $0 \leq l' \leq l$, PAS pa izračunamo iz*

$$PAS(D, l', l) = AS(e_{l'}, e_{l'+1}, ..., e_l). \tag{4.3}$$

Definicija 4.4 *Algebraični podpis n-grama (NAS) je specifična oblika delnega algebraičnega podpisa, pri čemer je dolžina med dvema odmikoma l enolično določena z velikostjo n-grama v spremenljivki n in kjer velja $l \geq n-1$,*

$$NAS(D, l) = PAS(D, l-n+1, l). \tag{4.4}$$

Če se po deskriptorju premaknemo za eno mesto in poznamo prejšnjo vrednost kumulativnega algebraičnega podpisa ali algebraičnega podpisa n-grama, si lahko računanje olajšamo z Definicijo 4.5.

Definicija 4.5 *Če poznamo predhodno vrednost kumulativnega algebraičnega podpisa, lahko naslednjega izračunamo po enačbi*

$$CAS(l) = CAS(l-1) + e_l \cdot \alpha. \tag{4.5}$$

Prav tako lahko izračunamo trenutni algebraični podpis n-grama z

$$NAS(l) = \frac{NAS(l-1) - e_{l-n}}{\alpha} + e_l \cdot \alpha^{n-1}. \tag{4.6}$$

Pri točnem iskanju smo se opirali na pomembno lastnost kumulativnega algebraičnega podpisa iz Definicije 4.6.

Definicija 4.6 *Za* $0 \leq l' < l$ *je možno poljubno kumulativno vsoto izračunati po enačbi*

$$CAS(l) = CAS(l') + \alpha_{l'+1} PAS(l'+1, l). \tag{4.7}$$

Poglavje 5

Tehnični vidik

V Poglavju 3 smo si ogledali opise funkcionalnosti, ki so vidne uporabniku, v okviru tehničnega vidika pa si bomo najprej pogledali programsko opremo, potrebno za postavitev in izvajanje naše aplikacije na strežniku. Nato si bomo ogledali uporabljene tehnologije in obstoječe knjižnice. Na koncu si bomo pogledali tudi podroben opis delovanja naše realizacije.

5.1 Potrebna programska oprema

5.1.1 Spletni strežnik Apache Tomcat 7

Ker smo želeli, da bi bila naša aplikacija dostopna na daljavo oziroma preko svetovnega spleta, smo morali predhodno namestiti spletni strežnik, ki bo sprejemal zahteve uporabnikov in jim odgovarjal z dokumenti HTML. Za opravljanje te naloge smo izbrali Apache Tomcat 7 [14], ki je odprtokoden, brezplačen in omogoča izvajanje programskega jezika Java v spletnem strežniku HTTP.

5.1.2 JVM

Za poganjanje strežnika in programske kode, napisane v jeziku Java, smo potrebovali virtualni stroj Java (angl. Java Virtual Machine – JVM). Tej

zahtevi smo zadostili z namestitvijo razvojnega kompleta Java (angl. Java Development Kit – JDK) [15].

5.1.3 Eclipse

Eclipse [16] je integrirano razvijalno okolje, namenjeno v prvi vrsti pisanju aplikacij v Javi. Uporabljali smo ga za razvijanje naše aplikacije in poganjanje spletnega strežnika Apache Tomcat.

5.2 Uporabljene tehnologije

5.2.1 HTML5

HTML (angl. HyperText Markup Language) [17] je oblikovni jezik, namenjen izgradnji spletnih strani. HTML5 pa je najnovejši standard za HTML. Da lahko brskalnik uspešno sestavi izgled strani, morajo biti uporabljene veljavne in pravilno ugnezdene značke. Vsak dokument HTML5 vsebuje značke za glavo (angl. head) in telo (angl. body). V glavi je pomembna značka za naslov (angl. title), telo pa je deljeno na posamezne dele z značkami `div`. Obstajajo še drugi načini delitve telesa, vendar je ta način najpogostejši. Izsek iz kode 5.1 je primer uporabe najosnovnejših elementov za sestavo spletne strani v HTML5.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Naslov</title>
</head>

<body>
Vsebina dokumenta...
</body>
</html>
```

Izsek iz kode 5.1: Primer preprostega dokumenta HTML5.

5.2.2 Java

Java [18] je objektno orientiran programski jezik. Temelji na razredih in je narejen po principu, da ima čim manj odvisnosti od implementacije.

5.2.3 JavaScript

JavaScript [19] je programski jezik, ki ga lahko vstavimo med značke v dokumentu HTML. Posebna prednost tega jezika je, da teče na strani odjemalca in ga izvaja brskalnik. S tem se razbremeni delo spletnega strežnika. Z njim lahko pišemo funkcije, ki lahko vplivajo na elemente HTML. Posebno uporaben je v kombinaciji s knjižnico jQuery [20], ki nam še dodatno lajša programiranje in oblikovanje aplikacij.

5.2.4 CSS

CSS (angl. Cascading Style Sheets) [21] je jezik za oblikovanje stila v dokumentu HTML. Nudi možnost oblikovanja posameznih elementov HTML. Z njim se lahko tudi izognemo večkratnemu definiranju oblike za isti element.

Ponavadi hranimo obširnejša oblikovna pravila v svoji datoteki CSS, možno pa jih je vključiti tudi neposredno v dokument HTML. Za oblikovanje elementa HTML moramo pred oblikovnimi pravili navesti ime elementa. Če želimo, da pravilo velja samo za določen element, pa za znakom # pripišemo še njegovo enolično oznako. Pravila so v nadaljevanju navedena znotraj zavitih oklepajev.

5.2.5 MusicXML

MusicXML [22] je eden najbolj znanih formatov za deljenje notnih zapisov. Odlikuje ga velika razširjenost uporabe, saj ga podpira večina sodobnih programov za ustvarjanje glasbe, kot so na primer Finale in Sieblius. Uporablja se že vse od leta 2004, ko so ga razvili pri podjetju Recordare LLC. Do danes je doživel že več izboljšav. Trenutna najnovejša razlličica je 3.0. Sam format, tako kot že ime pove, temelji na podobni strukturi kot XML.

5.3 Uporabljene obstoječe knjižnice

Pri realizaciji naše spletne aplikacije smo se opirali tudi na nekaj že obstoječih knjižnic, posameznih razredov in funkcij.

5.3.1 GaloisField

Za računanje v Galoisovem polju smo uporabili `GaloisField.java` [23]. V tem razredu so metode, ki nam omogočajo uporabo osnovnih operacij, kot so seštevanje in množenje, ki smo jih potrebovali pri računanju algebraičnih podpisov.

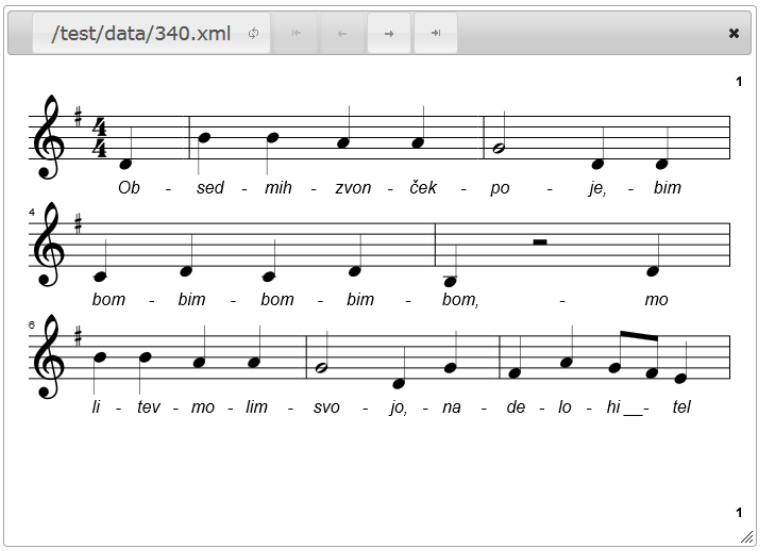
5.3.2 AudioSynth

Za proizvajanje zvokov na klaviaturi smo uporabili dinamični oscilatorski zvočni sintetizator iz skripte `audisynth.js`, ki deluje na podlagi uporabe

zvočnih elementov HTML5. Za grafično oblikovanje klaviature pa smo uporabili pripadajočo skripto `audiosynth.view.js` [24].

5.3.3 HTML5 MusicXML Viewer

Za predogled posameznega dokumenta MusicXML v rezultatih iskanja smo uporabili HTML5 MusicXML Viewer [25] in njegovo pripadajočo knjižnico `score-library`, napisano v jeziku JavaScript. Ta knjižnica nam omogoča, da pretvorimo datoteko MusicXML v notni prikaz, ki se izriše na elementu HTML5 `canvas`, ki si ga lahko ogledujemo na poljubnem brskalniku. Knjižnica ima sicer še določene pomanjkljivosti in je v stanju razvoja. Poleg tega notne predstavitve niso vedno točne. Kljub temu smo se odločili za uporabo te knjižnice, ker uporabniku vseeno nudi zadostno vizualno predstavo o vsebini posameznega dokumenta MusicXML pa tudi zaradi pomanjkanja boljših alternativ. Prikaz, ki ga nudi, si lahko ogledamo na Sliki 5.1.



Slika 5.1: Prikaz notnega zapisa.

5.3.4 VexTab in VexFlow

Za grafično notno predstavitev trenutno vnesene iskalne poizvedbe in prikaza ujemajočega izseka v rezultatih iskanja smo uporabili VexFlow [26] in VexTab [27]. VexFlow umogoča izris not na notnem črtovju s pomočjo jezika JavaScript in HTML5. VexTab, ki deluje na istih osnovah, pa se uporablja za hitro pisanje in spreminjanje notnih zapisov, ki jih potem prikaže na notnem črtovju.

5.4 Opis delovanja

5.4.1 Zajem podatkov

Potrebne podatke za izgradnjo indeksa smo iz datotek MusicXML najprej izluščili s pomočjo razčlenjevalnika DOM. Izsek iz kode 5.2 prikazuje elemente dokumenta MusicXML, ki nosijo podatke, ki smo jih potrebovali.

```
<note>
  <pitch>
    <step>C</step>
    <alter>0<alter>
    <octave>4</octave>
  </pitch>
  <duration>4</duration>
  <voice>1</voice>
</note>
```

Izsek iz kode 5.2: Splošen zapis podatkov za eno noto v dokumentu MusicXML.

Prikazani elementi skupaj predstavljajo eno noto, in sicer vsebujejo podatke o imenu note, oktavi, možnih poltonih, trajanju note in številki glasu, ki mu posamezna nota pripada. Podatke o imenu note, poltonih, oktavi smo združili v eno numerično vrednost, ki enolično določa omenjene podatke in

ji bomo rekli višina tona. Točne preslikave si lahko pogledamo v Tabeli 5.1. Pavz in ostalih glasbenih okraskov nismo vključili v svoj zajem podatkov. Prav tako smo izpustili morebitne istočasne tone ali akorde z enako številko glasu in upoštevali le en osnoven ton v določenem trenutku na posamezni glas.

Oktava/Ton	C	C#/Db	D	D#/Eb	E	F	F#/Gb	G	G#/Ab	A	A#/Bb	B
0	1	2	3	4	5	6	7	8	9	10	11	12
1	13	14	15	16	17	18	19	20	21	22	23	24
2	25	26	27	28	29	30	31	32	33	34	35	36
3	37	38	39	40	41	42	43	44	45	46	47	48
4	49	50	51	52	53	54	55	56	57	58	59	60
5	61	62	63	64	65	66	67	68	69	70	71	72
6	73	74	75	76	77	78	79	80	81	82	83	84
7	85	86	87	88	89	90	91	92	93	94	95	96
8	97	98	99	100	101	102	103	104	105	106	107	108
9	109	110	111	112	113	114	115	116	117	118	119	120

Tabela 5.1: Pripadajoče vrednosti višin tonov za posamezne note.

Za možna trajanja posameznih not smo izbrali le nekaj najbolj pogostih. Vsa preostala možna trajanja smo med zajemom zaokrožili k najbližjim vrednostim trajanj not iz Tabele 5.2. To smo storili iz razloga, ker nismo želeli narediti iskanja z ritmom preveč občutljivega in posledično zapletenega za uporabnika. Uporabniku bi bilo namreč težko vnesti točen ritem, ki ga želi najti, saj bi vsaka najmanjša napaka vnosa trajanja pomenila, da ustreznega zapisa ne bi našel.

Vrsta note	Številska vrednost
Dvaintridesetinka	1
Dvaintridesetinka s piko	2
Šestnajstinka	3
Šestnajstinka s piko	4
Osminka	5
Osminka s piko	6
Četrtinka	7
Četrtinka s piko	8
Polovinka	9
Polovinka s piko	10
Celinka	11
Celinka s piko	12

Tabela 5.2: Možna trajanja note in pripadajoče številske vrednosti.

Tako obdelane pare vrednosti višin tonov in trajanj smo po vrsti shranjevali v sezname (angl. ArrayList). V kateri seznam se doda določen par, pa je bilo odvisno od imena dokumenta in številke glasu, ki jima par pripada. En celoten seznam je ekvivalenten enemu deskriptorju posameznega glasu v določenem dokumentu. Vse skupaj smo hranili v zgoščevalnem slovarju (angl. HashMap), kjer je bil ključ sestavljen iz imena dokumenta in številke glasu. Pripadajoča vrednost pa je vsebovala ustrezen seznam parov višin in trajanj.

Poleg teh podatkov, ki so nam predstavljali vsebino, smo zajeli še morebitno prisotnih podatkov o naslovih skladb, opusov in imenih avtorjev v posameznem dokumentu MusicXML. Te podatke smo pridobili iz elementov MusicXML, ki jih prikazuje Izsek iz kode 5.3, in jih shranili v seznam, v katerem je vsak element vseboval podatke iz drugega dokumenta.

```
<work>
  <work-number>D. 911</work-number>
    <work-title>Winterreise</work-title>
</work>
<movement-number>22</movement-number>
<movement-title>Mut</movement-title>
<identification>
  <creator type="composer">Franz Schubert</creator>
  <creator type="poet">Wilhelm Muller</creator>
</identification>
```

Izsek iz kode 5.3: Primer elementov MusicXML, ki nosijo podatke o naslovu skladbe, opusu in avtorjih.

Tako seznam s podatki o naslovih in imenih kot zgoščevalni slovar z vsebino smo nato zapisali v ločeni datoteki za poznejšo uporabo pri izgradnji indeksa, iskanju in prikazu rezultatov.

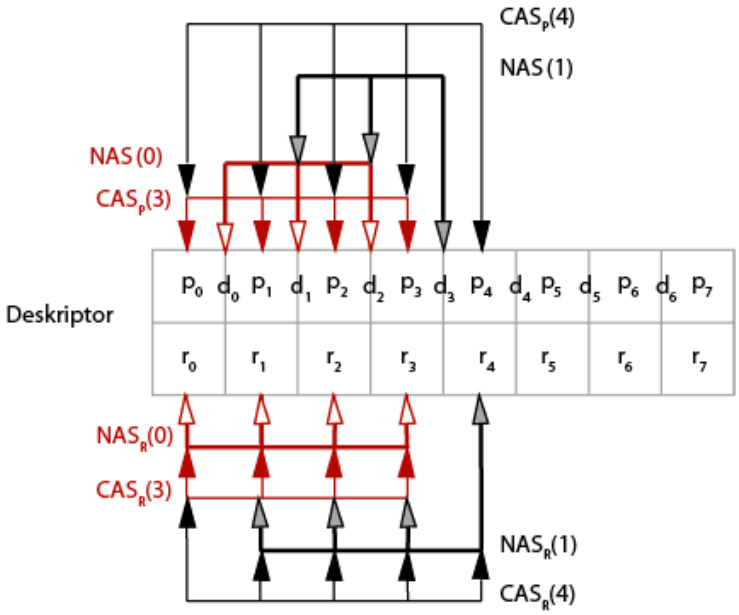
5.4.2 Izgradnja indeksa

Za izgradnjo indeksa smo najprej naložili datoteko, ki vsebuje zgoščevalni slovar s seznami, ki predstavljajo deskriptorje posameznega dokumenta in glasu, nazaj v delovni spomin. Posamezni seznam smo razdelili na več delov, ki jim rečemo n-grami, z velikostmi $n = 3$, $n = 4$, $n = 5$ in $n = 6$. To smo naredili s pomočjo pomičnega okna, ki je bil enak trenutno izbrani velikosti n-grama in ki smo ga pomikali za en element naprej od začetka do konca seznama. Pred prvim pomikom okna smo za prvi n-gram izračunali vrednost NAS po Definiciji 4.4, kjer smo za elemente e vzeli vrednosti razdalj med sosednimi višinami tonov v n-gramu in $\alpha = 2$. To smo ponovili še z $\alpha = 3$ in $\alpha = 5$ in dobili skupaj tri vrednosti. Dobljene vrednosti smo nato združili z bitno konkatenacijo 5.1.

$$HD(i) = AS_{\alpha_5}.AS_{\alpha_3}.AS_{\alpha_2} mod L \tag{5.1}$$

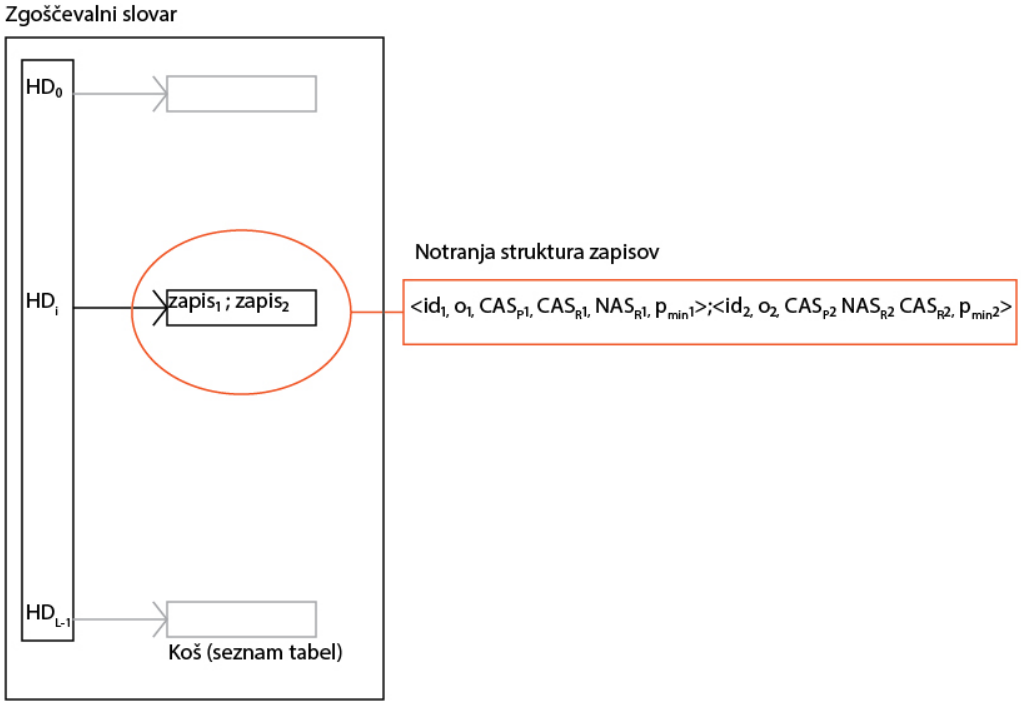
Vrednost L v enačbi omejuje število bitov, iz katerih je lahko sestavljena posamezna vrednost HD. Tako združena vrednost $HD(i)$ nam je predstavljala vrednost indeksa, kateremu trenutni n-gram pripada oziroma vrednost enega ključa v našem zgoščevalnen slovarju. Posameznemu ključu pa pripada seznam s shranjenimi zapisi n-gramov z enako vrednostjo $HD(i)$.

Nato smo izračunali še posamezne vrednosti, ki sestavljajo zapis izbranega n-grama, tako da smo za n-gram izračunali vrednost CAS po Definiciji 4.2, pri čemer smo za elemente e vzeli vrednosti višin tonov v izbranem n-gramu in $\alpha = 2$. Dobljeno vrednost poimenujmo CAS_p. Hkrati smo računali po Definiciji 4.4 tudi NAS za ritem, kjer smo za elemente e vzeli trajanja tonov v n-gramu in $\alpha = 2$ in dobili vrednost, ki jo poimenujmo NAS_R. Ravno tako smo za ritem računali tudi CAS in dobili vrednost CAS_R po isti definiciji, kot smo to storili za izračun CAS_p, le da samo za vrednosti e spet vzeli trajanja. Beležili smo tudi, na katerem odmiku o se n-gram nahaja, najnižjo višino tona p_{min} v n-gramu in številko id, ki pove, kateremu dokumentu in številki glasu posamezni n-gram pripada. Vse te vrednosti sestavljajo en zapis enega n-grama. Ob vseh nadaljnjih pomikih okna smo ponovili postopek, z razliko, da smo za izračun posameznih vrednosti NAS in CAS uporabili Definicijo 4.5. Ta nam prihrani ponovno preračunavanje predhodno že delno izračunanih vrednosti CAS in NAS. Katere vrednosti deskriptorja posamezna vrednost CAS in NAS zajema, si lahko ogledamo na Sliki 5.2, kjer so vizualizirane te vrednosti za prva dva n-grama.



Slika 5.2: Primer deskriptorja dolžine 8 z velikostjo n-grama $n = 4$, ki vsebuje elemente višin tonov p in trajanj tonov r. Vmesni elementi d predstavljajo razlike med sosednimi višinami tonov in jih izračunamo posebej, ker niso vsebovane v samem deskriptorju. Rdeče vrednosti pripadajo prvemu n-gramu, črne pa drugemu.

Celoten postopek smo ponovili za vsak nadaljnji seznam v zgoščevalnem slovarju, ki smo ga dobili pri zajemu podatkov. Tako pridobljeni novi indeksni zgoščevalni slovar smo nato shranili v datoteko za poznejšo rabo pri iskanju. Izgled strukture našega indeksa in zapisov n-gramov si lahko ogledamo na Sliki 5.3.



Slika 5.3: Zgoščevalni slovar hrani vrednosti indeksa v posameznih ključih $HD(i)$ in zapise n-gramov v pripadajočih seznamih. Podatki so v posameznem zapisu hranjeni v tabeli.

5.4.3 Točno iskanje

Za točno iskanje potrebujemo indeksni zgoščevalni slovar, ki smo ga sestavili v Podpoglavju 5.4.2. Za opis rezultatov iskanja potrebujemo tudi seznam s podatki o naslovih skladb in imenih avtorjev, ki smo ga dobili pri zajemu podatkov v Podpoglavju 5.4.1. Še ena pomembna spremenljivka, ki jo moramo poznati, je velikost n-grama. Ta mora biti enaka velikosti n-grama, uporabljeni pri izgradnji indeksnega zgoščevalnega slovarja.

Da sploh lahko pričnemo z iskanjem, potrebujemo v prvi vrsti ustrezen iskalni niz, ki vsebuje višine in trajanja tonov. Poleg tega vsako iskanje potrebuje še podatke o stanju parametrov, kot so iskanje z ritmom, transponiranje in iskanje samo po začetkih zapisov v zbirki. Ko je vključeno

iskanje z ritmom, se poleg iskanja po melodiji dodatno preverja še ujemanje ritma. Možnost transponiranega iskanja naredi iskanje neobčutljivo na dejansko višino tona. Važne so le razdalje med višinami sosednjih tonov.

Preden začnemo z dejanskim iskanjem, najprej razdelimo iskalni niz, ki vsebuje višine in trajanja, na začetni n-gram $S1$, končni n-gram $S2$ in del med njima, ki mu bomo rekli vmesni niz SV, katerega sestavljajo vse vrednosti višin in trajanj, ki niso del prvega ali zadnjega n-grama. Tako za n-gram $S1$ kot $S2$ izračunamo pripadajoče vrednosti HD in NAS_R, določimo odmik o glede na lokacijo v iskalnem nizu in najnižjo višino v posameznem n-gramu p_{min}. Omenjene vrednosti izračunamo na enak način, kot smo to storili pri postopku izgradnje indeksa v Podpoglavju 5.4.2, z razliko, da tu računamo ti vrednosti samo za dva posamezna n-grama in nimamo opravka s celotnim deskriptorjem.

Točno iskanje ima torej to posebnost, da za ugotavljanje ujemanja potrebujemo le začetni in končni n-gram iskalnega niza. To naredimo tako, da najprej v ključih indeksnega zgoščevalnega slovarja poiščemo vrednosti HD_{S1} od n-grama S1 in HD_{S2}, ki pripada n-gramu S2. Če ti vrednosti ključa obstajata v našem indeksnem zgoščevalnem slovarju, dobimo za vsako vrednost en seznam z zapisi n-gramov, ki pripadajo vrednosti HD_{S1} in drugi seznam zapisov, ki pripadajo HD_{S2}. Ta dva seznama torej vsebujeta potencialne kandidate, ki bi se lahko ujemali z n-gramom S1 ali S2 iz iskalnega niza. Če bodisi en ali drugi seznam ne obstaja ali pa ne obstaja noben, z iskanjem zaključimo že tu. To posledično pomeni, da ni bilo ustreznih ujemanj v zbirki glede na izbran iskalni niz.

Drugače pa nadaljujemo z izključevanjem elementov obeh seznamov nabora kandidatov, tako da gremo po obeh seznamih in sestavimo vse možne kombinacije parov elementov, ki predstavljajo zapise kandidatov za S1 in S2. Zanimajo nas le zapisi parov S1 in S2, ki so iz istega dokumenta in se pojavijo v istem glasu. Veljati mora tudi, da sta n-grama enako oddaljena eden od drugega kot prvotna n-grama S1 in S2 iz iskalnega niza. Poleg tega se morajo vrednosti p_{min} v primeru, da iščemo točno določene višine tonov

ujemati. Če je tem pogojem zadoščeno, elementa para shranimo v ločena seznama, enega za elemente S1 in enega za elemente S2. Elementa S1 in S2 iz istega para sta shranjena v seznamih na istem mestu. S tem overimo, kateri dokumenti in številke glasov vsebujejo n-grame, ki so enaki začetnemu in končnemu n-gramu in je med njima enaka razdalja. Ne vemo pa še, če se ujema vmesni niz SV.

Problem nastane, ker vrednost CAS, ki pripada zapisu n-grama S2 iz seznama, ni nujno enaka vrednosti CAS n-grama S2 iz iskalnega niza, tudi v primeru, če se vmesni niz ujema. To se zgodi zato, ker se niz, ki ga iščemo, nujno ne nahaja na začetku deskriptorja posameznega dokumenta in glasu. Zato uporabimo Definicijo 4.6, ki smo jo omenili v Podpoglavju 4.2. Za uspešen izračun potrebujemo CAS_P posameznega zapisa n-grama S1 iz seznama in PAS_P celotnega iskalnega niza razen dela, ki pripada n-gramu S1. Če se tako izračunan CAS_P ujema s CAS_P zapisa n-grama S2 v seznamu sledi, da smo dobili ujemanje in je vmesni del enak za trenutni par S1 in S2.

Poudariti je treba, da sta kumulativni in delni podpis občutljiva na višino tonov in v primeru, da iščemo tudi transponirane višine, je potrebno višine, vsebovane v iskalnem nizu ustrezno prilagoditi pred računanjem vrednosti PAS_P. To naredimo preprosto tako, da pogledamo, katera je najmanjša višina v n-gramu S2 v zapisih in ustrezno prestavimo višine tonov višje ali nižje, in sicer za razliko med najmanjšo višino tona v n-gramu S2 iz zapisa in S2 iz iskalnega niza. Prav tako je potrebno paziti v primeru, da se išče tudi po ritmu. Postopek je tu enak kot za izračunavanje CAS_P, le da gre tu CAS, ki ga računamo po trajanjih in ne višinah tonov. Potrebujemo torej CAS_R posameznega zapisa n-grama S1 iz seznama in nato izračunamo še PAS_R istega dela kot pri PAS_P, z razliko, da za računanje le-tega spet vzamemo trajanja. Izračunani CAS_R primerjamo z vrednostjo CAS_R zapisa n-grama S2 v seznamu in če se ujemata, predvidevamo, da se ritem ujema.

Rezultate shranjujemo v seznam, v katerem hranimo podatke o imenu datoteke MusicXML in številki glasu, v kateri je prišlo do ujemanja, naslovu skladbe, opusa, imenu avtorja, odmiku v deksriptorju podatkov, na katerem

se je začelo in končalo ujemanje.

5.4.4 Približno iskanje

Pri približnem iskanju, kot že samo ime pove, se iskalni niz ne rabi popolnoma ujemati z rezultati iskanja. Na kratko bi lahko način tega iskanja povzeli kot: več n-gramov, kot imata dva niza skupnih, bolj sta si podobna.

Za začetek iskanja potrebujemo enake vhodne podatke in parametre kot pri točnem iskanju. Edini dodatni parameter je največja dovoljena razdalja v ujemanju n-gramov niza iskanja ter n-gramov iz zapisov, ki jih primerjamo. Ta je potreben, da se določi največjo dovoljeno razdaljo v ujemanju n-gramov, ki je še dovoljena, da določen del iz dokumentov v zbirki še spada med rezultate iskanja.

Razlike od točnega iskanja se v postopku začnejo pri tem, da namesto razdelitve na končni, začetni n-gram in srednji niz, razdelimo celotni iskalni niz na vse možne n-grame, ki jih ta vsebuje. Za te izračunamo posamezne vrednosti HD in vrednosti NAS_R.

Sedaj za vse tako dobljene vrednosti $HD(i)$ v indeksnem zgoščevalnem slovarju poiščemo pripadajoče sezname. Nato gremo po vseh izbranih seznamih in premikamo zapise n-gramov, ki pripadajo istemu dokumentu in številki glasu, v ločene sezname. Te sezname uredimo tako, da n-grame razvrstimo po vrednosti odmika o. Če je katera vrednosti odmika nezasedena, v seznam vstavimo prazen element, drugače pa mora vsebovati vrednosti HD, NAS_R in p_{min} posameznega n-grama. Tako pripravljene sezname je sedaj potrebno pregledati enega za drugim.

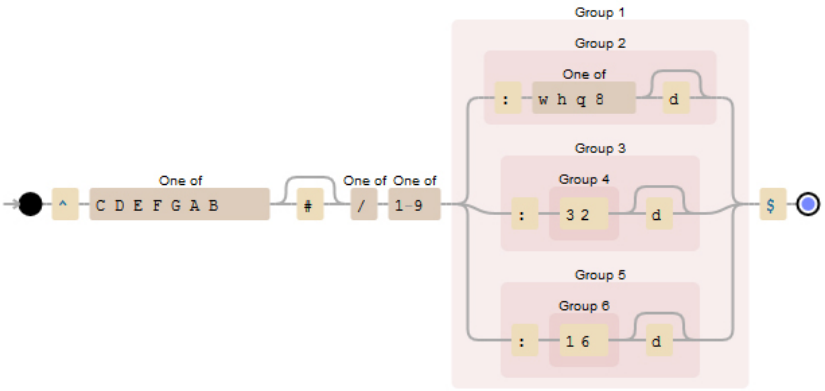
Posamezni seznam pregledamo za ujemanje tako, da uvedemo pomično okno velikosti $2 * m - 2 * n + 2$, pri čemer je m dolžina iskalnega niza, n pa velikost n-grama. Preden ga pomaknemo za eno mesto po seznamu, moramo pregledati same vrednosti n-gramov, ki jih trenutno vsebuje pomično okno. Pred pregledovanjem okna privzamemo, da je začetna razdalja velikosti $m - n + 1$, kar je enako številu vseh možnih n-gramov v iskalnem nizu. Nato po vrsti glede na odmik preverjamo vrednost HD zapisov n-gramov znotraj

okvirja in za vsako ujemanje z n-grami iz iskalnega niza zmanjšamo razdaljo za ena, če n-gram ni v iskalnem nizu pa se razdalja poveča za ena. Tu se predpostavi, da se n-grama ujemata, če imata enako vrednost HD, dejanske pozicije glede na lokacijo n-grama v iskalnem nizu se ne preverja. Prav tako se vsak prazen element seznama šteje kot n-gram, ki ga iskalni niz ne vsebuje in prav tako poveča razadljo. V primeru, da ne iščemo transponirane melodije moramo biti pri primerjanu pozorni tudi na vrednost P_{min}, ki se mora ujemati pri primerjavi n-gramov. Ravno tako mora biti enak NAS_R n-grama, če iščemo še po ritmu.

Med primerjanjem teh n-gramov v oknu sledimo tudi vrednosti razdalje in končno pozicijo, kjer je ta najmanjša v trenutnem celotnem oknu, shranimo. Prav tako shranimo začetno pozicijo okna. Celoten postopek ponovimo za vse premike pomičnega okna. Ko okno pride do konca posameznega seznama zapisov n-gramov, izmed vseh ujemanj ohranimo le tisto z najmanjšo razdaljo ujemanj v n-gramih. Če je ta vrednost manjša ali enaka dovoljeni, potem shranimo razdaljo, začetno in končno pozicijo tega ujemanja ter v katerem dokumentu in številki glasu se nahaja. Celoten postopek s pomičnim oknom ponovimo za vsak urejen seznam zapisov n-gramov iz posameznega dokumenta in glasu. Pridobljene rezultate, nato še posortiramo po posameznih razdaljah, kar je uporabno pri izpisu in prikazu rezultatov.

5.4.5 Vnos iskanja in tvorjenje rezultatov iskanja

Posamezni toni, ki jih vnašamo ob igranju klaviature ali tekstovno, se hranijo v tekstovnem elementu HTML `input`, ki hrani tekstovni niz poizvedbe. Ob vsaki spremembi niza v tem elementu se sproži funkcija, ki najprej preveri ustreznost vnosa. Pravilnost vnosa se preveri z regularnimi izrazi (angl. Regular expression - regex) [28]. Na Sliki 5.4 in Sliki 5.5 sta predstavljena dva glavna izraza, ki smo ju uporabili za preverjanje. Pregleduje se vsako noto posebej in sicer tako, da se predvideva, da so posamezni zapisi za noto v nizu ločeni z vejico ali presledki.



Slika 5.4: Regularni izraz za preverjanje posamezne note z definiranim trajanjem.

Slika 5.5: Regularni izraz za preverjanje posamezne note s privzetim trajanjem.

Po opravljenem preverjanju niza se upošteva samo pravilno zapisane note, ostalo pa se izpusti. Pravilno zapisane note se izrišejo na notnem črtovju v elementu HTML5 `canvas` s pomočjo knjižnice VexFlow. Če želimo te note predvajati, se sproži tudi pretvorba v ustrezen format, ki ga sprejema skripta `audisynth.js`, s katero to zaporedje not lahko predvajamo. Omenjeno skripto je bilo potrebno prilagoditi, da je podpirala naša izbrana trajanja tonov.

Ves čas se hranijo tudi pretvorbe trenutno vnešenih not v ustrezne številske vrednosti višin in trajanj not v skritem elementu HTML `input`. Skriti element z višinami in trajanji not in ostale vhodne parametre za iskanja hranimo v elementu HTML `form`. Ti podatki se ob pritisku na gumb za iskanje

pošljejo strežniku po metodi `POST`. Pred dejanskim pošiljanjem se s funkcijo JavaScript preveri, če je število not v skritem nizu večje ali enako velikosti n-gramov, ki so izbrani za iskanje.

Strežnik nato prejme podatke o parametrih in iskalnem nizu ter ustrezno izvede eno od dveh možnih iskanj. Rezultati točnega iskanja se nato še filtrirajo, tako da se primerja ujemajoče note iz iskalnega niza z notami ki naj bi se ujemale v posameznem rezultatu. To je potrebno izvesti, saj lahko pride do lažnih ujemanj, posebno zato ker se vmesni niz med začetnim in končnim n-gramom preverja le s kumulativnim podpisom. Tako filtrirane rezultate se nato zapiše v tabelo v dokumentu HTML. K vsakemu rezultatu se doda še povezavo do dokumenta MusicXML, ki mu pripada, in predogleda celotnega notnega zapisa s pomočjo ogledovalnika dokumentov MusicXML v HTML5. Ujemajoči del se tudi izriše posebej v elementu HTML `canvas` s pomočjo knjižnice VexTab. Tako zgrajen dokument HTML strežnik nato pošlje uporabniku.

Poglavje 6

Preizkus realizirane rešitve na testnih podatkih

Pri preizkusu naše realizacije na testnih podatkih nas je poleg same hitrosti izvedbe iskanja najbolj zanimala časovna razlika med točnim in približnim iskanjem in kaj najbolj vpliva nanju.

Tabela 6.1 prikazuje podrobnosti o naši uporabljeni zbirki glasbenih zapisov. V Tabeli 6.2 vidimo, da se indeksne datoteke razlikujejo glede na število vrednosti indeksa. Večji kot je n-gram, več različnih vrednosti vsebuje. Za največjo možno vrednost indeksa za to zbirko smo vzeli vrednost 4095. Vrednost indeksa je bila zato lahko največ 12-bitna.

Zbirka	Velikost zbirke	Število datotek	Največja dovoljena velikost vrednosti indeksa
Šolska zbirka	160 MB	4295	4095

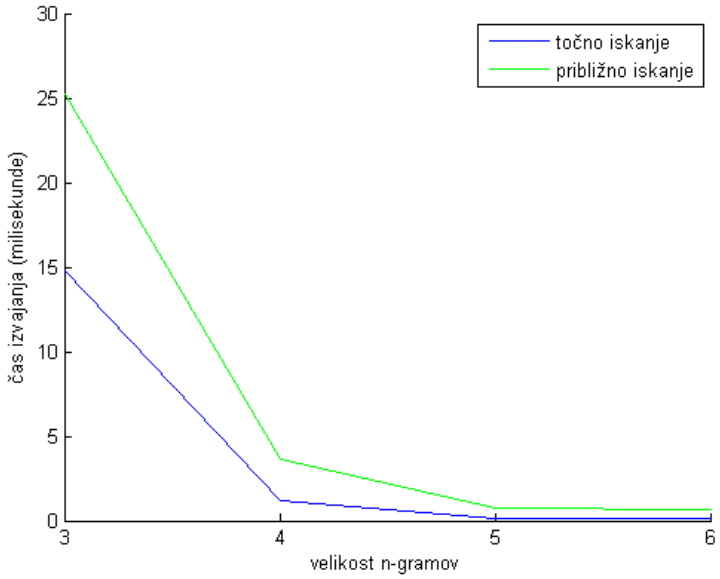
Tabela 6.1: Značilnosti uporabljene zbirke za testiranje.

Najprej smo poskusili z iskalno poizvedbo dolžine osmih tonov. Glede na to, da točno iskanje potrebuje le začetni in končni n-gram, smo predvideli, da bo to iskanje časovno manj zahtevno od približnega iskanja, ki potrebuje izračun in iskanje vseh n-gramov poizvedbe. Prav tako smo predvideli, da bo na obe vrsti iskanja vplivala velikost n-gramov, ki se uporablja pri iskanju in indeksiranju. Večji n-gram pomeni večjo raznolikost in manj enakih n-

Velikost n-gramov	Dolžina indeksnega slovarja
3	481
4	2497
5	4006
6	4094

Tabela 6.2: Dolžina indeksnega slovarja pri določeni velikosti n-gramov.

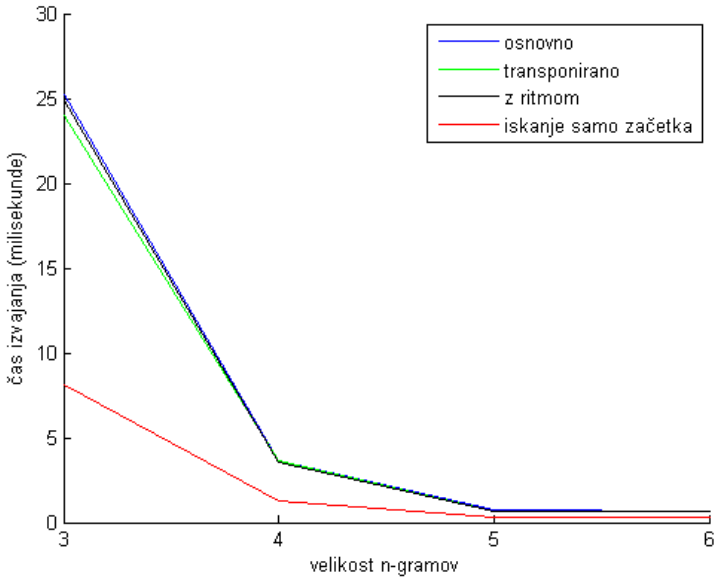
gramov, ki jih moramo pri iskanju pregledati. Naše meritve so potrdile obe domnevi. Na Sliki 6.1 vidimo, da je bilo točno iskanje hitrejše od približnega pri vseh velikostih n-gramov. Večji n-grami pa so tudi zelo skrajšali čas iskanja.



Slika 6.1: Graf, ki prikazuje dve vrsti iskanja, in sicer čas iskanja v odvisnosti od velikosti n-gramov, uporabljenih pri iskanju.

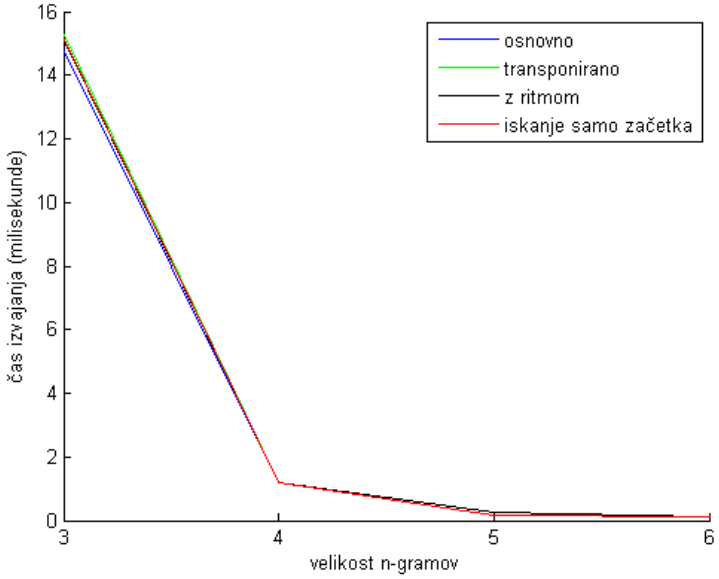
V nadaljevanju nas je zanimalo, kateri parametri najbolj vplivajo na dolžino iskanja. Glede na strukturo naše realizacije smo predvidevali, da je

transpozicija in ritem ne bosta občutno spremenila, saj sta v večini vključena že v obe vrsti osnovnega iskanja. N-grami so sestavljeni iz razdalj med toni, kar pomeni, da že osnovno iskanje zajame vse morebitne transpozicije. Razlika je le v tem, da jih ne prikaže kot rezultate iskanja. Predvidevali smo, da bo imel največji vpliv na dolžino približnega iskanja parameter iskanja samo začetka zapisov. Ta lahko občutno zmanjša potrebno število n-gramov, ki jih mora posamezno iskanje pregledati. Pri točnem iskanju nismo predvideli občutnih sprememb, ker ta parameter temu služi le kot eden od pogojev določevanja ustreznosti n-grama in ne omejuje števila n-gramov, ki jih je potrebno pregledati v posameznem iskanju. Meritve so nam spet potrdile pravilnost obeh domnev. Pri približnem iskanju se je čas iskanja občutno skrajšal pri vseh velikostih n-gramov, kar je prikazano na grafu Slike 6.2.



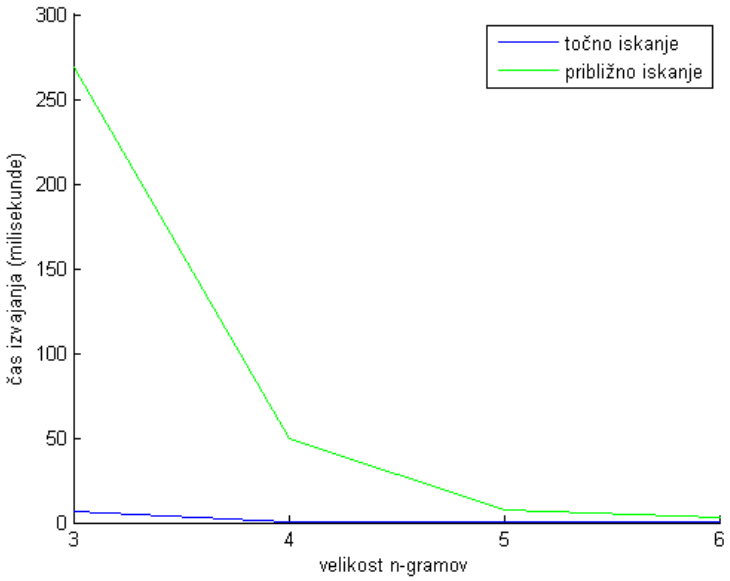
Slika 6.2: Graf, ki prikazuje vpliv parametrov na čas približnega iskanja v odvisnosti od velikosti n-gramov.

Pri točnem iskanju pa na grafu iz Slike 6.3 vidimo, da so dolžine časa iskanja ostajale približno enake.



Slika 6.3: Graf, ki prikazuje vpliv parametrov na čas točnega iskanja v odvisnosti od velikosti n-gramov.

Na koncu smo se namenili potrditi glavno prednost, ki naj bi jo imelo točno iskanje pred približnim, in sicer neodvisnost dolžine časa iskanja glede na dolžino poizvedovalnega niza. Ponovili smo meritve obeh vrst iskanj z nizom dvakratne velikosti niza, uporabljenega v vseh prejšnjih meritvah časov iskanj. Če primerjamo graf na Sliki 6.1 z grafom na Sliki 6.4, vidimo, da so vrednosti pri približnem iskanju občutno večje. Vrednosti točnega iskanje pa ostajajo enake, kar potrjuje neodvisnost dolžine časa iskanja od dolžine poizvedbe pri točnem iskanju.



Slika 6.4: Graf, ki prikazuje dve vrsti iskanja, in sicer čas iskanja v odvisnosti od velikosti n-gramov pri daljšem poizvedovalnem nizu.

Poglavje 7

Sklepne ugotovitve

Naš poskus realizacije spletne aplikacije za indeksiranja in iskanja po simboličnih glasbenih zbirkah se vsekakor lahko primerja z že obstoječimi aplikacijami.

Prav tako lahko potrdimo, da smo teorijo iz članka realizirali na zadovoljiv način. Dobili smo podobne meritve, kot jih dokumentira članek [30] po katerem smo realizirali našo spletno aplikacijo. Točno iskanje je učinkovito, ko imamo dolge iskalne nize in smo dokaj gotovi o naši pravilnosti vnešenega iskalnega niza. Za krajše iskalne nize pa lahko brez pomislekov izberemo približno iskanje, kjer zanemarljivo časovno izgubo odtehta dejstvo, da ta od nas ne zahteva točne melodije ali ritma, da dobimo željen rezultat.

Na tem mestu velja še enkrat poudariti, da naša trenutna realizacija ni primerna za zapise, ki vsebujejo večglasja oziroma notne zapise, ki nimajo ločenih posameznih glasov. Prav tako je zajem podatkov zbirke specifično prilagojen za zbirko datotek MusicXML, ki smo jo uporabljali pri testiranju.

V našo realizacijo smo poskušali vključiti tudi čim več obstoječih knjižnic za delo z notami in zvoki, kot so VexFlow, VexTab, AudioSynth ter HTML5 MusicXML Viewer, ki so bile prosto dostopne prek spleta. Te so nam služile zadovoljivo, vendar trenutno še niso povsem razvite. Pri knjižnicah VexFlow in VexTab je tako še kar nekaj funkcionalnosti v stanju razvoja. Ena od večjih je na primer možnost predvajanja prikazanega notnega zapisa. Tudi HTML5

MusicXML Viewer ima kar nekaj težav z branjem različnih elementov MusicXML. Sicer sta obe deležni stalnih popravkov, tako da lahko pričakujemo, da bosta v nekaj letih še veliko boljši in celoviti.

Možnih je tudi še nekaj izboljšav in sprememb naše spletne aplikacije. Trenutno se na primer za indeksiranje uporablja melodijo, ker smo predvidevali, da je to najboljša opisna lastnost za našo testno zbirko. Za zbirke, ki vsebujejo pretežno ritmične zapise, kot so na primer notni zapisi za tolkala, je mogoče dokaj lahko prilagoditi delovanje naše aplikacije tako, da zamenjamo razdalje med toni s trajanji in indeksiramo po njih. Prav tako je možna razširitev števila podatkov, ki jih hrani posamezni zapis n-grama. Dodamo lahko podatke o inštrumentu, tonaliteti, taktovskim načinu in drugih lastnostih, ki jih lahko ima notni zapis glede na glasbeno teorijo.

Literatura

[1] (2014) Neuma. Dostopno na:
http://www.neuma.fr/

[2] (2014) Peachnote. Dostopno na:
http://www.peachnote.com/

[3] (2014) Musipedia. Dostopno na:
http://www.musipedia.org/

[4] (2014) Melodycatcher. Dostopno na:
http://www.melodycatcher.com/

[5] (2014) Hymnary. Dostopno na:
http://www.hymnary.org/melody/search

[6] (2014) Folktunefinder. Dostopno na:
http://www.folktunefinder.com/

[7] (2014) Nzdl. Dostopno na:
http://www.nzdl.org/fast-cgi-bin/music/musiclibrary

[8] (2014) Dodososo. Dostopno na:
http://www.dodososo.com/

[9] (2014) Tunefind. Dostopno na:
http://trillian.mit.edu/ jc/cgi/abc/tunefind

[10] (2014) Themefinder. Dostopno na:
http://www.themefinder.org/

[11] (2014) Pitch Contour. Dostopno na:
http://en.wikipedia.org/wiki/Pitch_contour

[12] (2014) Parsons code. Dostopno na:
http://en.wikipedia.org/wiki/Parsons_code

[13] (2014) Finite field arithmetic. Dostopno na:
http://en.wikipedia.org/wiki/Finite_field_arithmetic

[14] (2014) Apache Tomcat. Dostopno na:
http://tomcat.apache.org/

[15] (2014) JDK. Dostopno na:
http://en.wikipedia.org/wiki/Java_Development_Kit

[16] (2014) Eclipse. Dostopno na:
http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/keplersr2

[17] (2014) HTML. Dostopno na:
http://en.wikipedia.org/wiki/HTML

[18] (2014) Java. Dostopno na:
http://en.wikipedia.org/wiki/Java_programming_language

[19] (2014) JavaScript. Dostopno na:
http://en.wikipedia.org/wiki/JavaScript

[20] (2014) jQuery. Dostopno na:
http://jquery.com/

[21] (2014) CSS. Dostopno na:
http://en.wikipedia.org/wiki/Cascading_Style_Sheets

[22] (2014) MusicXML. Dostopno na:
http://en.wikipedia.org/wiki/MusicXML

[23] (2014) GaloisField.java. Dostopno na:
https://svn.apache.org/repos/asf/hadoop/common/branches/HDFS-1073/mapreduce/src/contrib/raid/src/java/org/apache/hadoop/raid/GaloisField.java

[24] (2014) AudioSynth. Dostopno na:
http://keithwhor.github.io/audiosynth/

[25] (2014) HTML5 MusicXML Viewer. Dostopno na:
http://musicxml-viewer.com/

[26] (2014) VexFlow. Dostopno na:
http://www.vexflow.com/

[27] (2014) VexTab. Dostopno na:
http://www.vexflow.com/vextab/

[28] (2014) Regular Expression. Dostopno na
http://en.wikipedia.org/wiki/Regular_expression

[29] W. Bas de Haas, Frans Wiering, Remco C. Veltkamp, "A geometrical distance measure for determining the similarity of musical harmony", *International Journal of Multimedia Information Retrieval*, št. 3, zv. 2, str. 189–202, 2013.

[30] Camelia Constantin, Cédric du Mouza, Zoé Faget, Philippe Rigaux, "The melodic signature index for fast content-based retrieval of symbolic scores", v zborniku *The 12th International Society for Music Information Retrieval Conference*, Miami, USA, oktober 2011, str. 363-368.

[31] Kjell Lemström, Niko Mikkilä, Veli Mäkinen, "Filtering methods for content-based retrieval on indexed symbolic music databases", *Information Retrieval*, št. 1, zv. 13, str. 1–21, 2010.